

EEEEEEEEEEEEEE XXX XXX CCCCCCCCCCCCCC HHH HHH NNN NNN GGGGGGGGGGGGG
EEEEEEEEEEEEEE XXX XXX CCCCCCCCCCCCCC HHH HHH NNN NNN GGGGGGGGGGGGG
EEEEEEEEEEEEEE XXX XXX CCCCCCCCCCCCCC HHH HHH NNN NNN GGGGGGGGGGGGG
EEE XXX XXX CCC HHH HHH NNN NNN NNN GGG
EEE XXX XXX CCC HHH HHH NNN NNN NNN GGG
EEE XXX XXX CCC HHH HHH NNN NNN NNN GGG
EEE XXX XXX CCC HHH HHH NNNNNNN NNN GGG
EEE XXX XXX CCC HHH HHH NNNNNNN NNN GGG
EEE XXX XXX CCC HHH HHH NNNNNNN NNN GGG
EEEEEEEEEEEE XXX XXX CCC HHHHHHHHHHHHHHHHH NNN NNN NNN GGG
EEEEEEEEEEEE XXX XXX CCC HHHHHHHHHHHHHHHHH NNN NNN NNN GGG
EEEEEEEEEEEE XXX XXX CCC HHHHHHHHHHHHHHHHH NNN NNN NNN GGG
EEE XXX XXX CCC HHH HHH NNN NNNNNNN GGG GGGGGGGGGGG
EEE XXX XXX CCC HHH HHH NNN NNNNNNN GGG GGGGGGGGGGG
EEE XXX XXX CCC HHH HHH NNN NNNNNNN GGG GGGGGGGGGGG
EEE XXX XXX CCC HHH HHH NNN NNN GGG GGG
EEE XXX XXX CCC HHH HHH NNN NNN GGG GGG
EEE XXX XXX CCC HHH HHH NNN NNN GGG GGG
EEE XXX XXX CCC HHH HHH NNN NNN GGG GGG
EEEEEEEEEEEEEE XXX XXX CCCCCCCCCCCCCC HHH HHH NNN NNN GGGGGGGGGGG
EEEEEEEEEEEEEE XXX XXX CCCCCCCCCCCCCC HHH HHH NNN NNN GGGGGGGGGGG
EEEEEEEEEEEEEE XXX XXX CCCCCCCCCCCCCC HHH HHH NNN NNN GGGGGGGGGGG

FILEID**EXCLIB

D 5

The image shows a 10x10 grid of numbers. The first four columns contain the sequence 8888, the next three columns contain 3333, and the last three columns contain 2222. Each row and column is a permutation of these four sequences.

EX

!MODULE exch\$library (IDENT = 'V04-000') = %TITLE 'Facility-wide library module'
BEGIN

* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
* ALL RIGHTS RESERVED.

* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
* TRANSFERRED.

* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
* CORPORATION.

* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.

++
FACILITY: EXCHANGE - Foreign volume interchange facility

ABSTRACT: BLISS Library for EXCHANGE facility

ENVIRONMENT: VAX/VMS User mode

AUTHOR: CW Hobbs , CREATION DATE: 1-July-1982

MODIFIED BY:

V03-002 CWH3002 CW Hobbs 12-Apr-1984
Add NOREMOTE, NOTSAMEDEV and RT11_DIRSIZE message codes.

--

```
! Include files:  
!  
! LIBRARY files:  
LIBRARY  
  'SYSSLIBRARY:LIB'          ! VMS operating system library  
;  
!  
! REQUIRE files:  
REQUIRE  
  'LIBS:EXCDEFS'            ! include the SDL definitions  
;  
!  
! Macros:  
!  
! Declare some macros as shorthand for the psect names  
MACRO  
  $global_rw = PSECT GLOBAL  = exch$rw_global (ADDRESSING_MODE (LONG_RELATIVE)); GLOBAL %  
;
```

```
; Declare some common data structure initialization macros  
MACRO  
    ! Define shorthand for a single initialized dynamic string desc  
    $dyn_str_desc          ! Static declaration  
        =  
        BLOCK [dsc$k_d_bln,BYTE]  
        PRESET ([dsc$b_class] = dsc$k_class_d,  
                 [dsc$b_dtype] = dsc$k_dtype_t,  
                 [dsc$w_length] = 0,  
                 [dsc$a_pointer] = 0')  
        %,  
  
    $dyn_str_desc_init (desci)      ! Run-time initialization  
        =  
        BEGIN  
        BIND  
            desc = (desci) : VECTOR [2, LONG],  
            tmpl = exch$go dyn_str_template : VECTOR [2, LONG];  
            desc [0] = .tmpl [0];  
            desc [1] = .tmpl [1];  
        END  
        %,  
  
    ! Define macro for a single initialized static string desc.  
    $stat_str_desc (L, A)          ! Static declaration  
        =  
        BLOCK [dsc$k_s_bln,BYTE]  
        PRESET( [dsc$b_class] = dsc$k_class_s,  
                 [dsc$b_dtype] = dsc$k_dtype_t,  
                 [dsc$w_length] = (L),  
                 [dsc$a_pointer] = (A)')  
        %,  
  
    $stat_str_desc_init (desci, L, A)  ! Run-time initialization  
        =  
        BEGIN  
        BIND  
            desc = (desci) : BLOCK [, BYTE];  
            desc [dsc$b_class] = dsc$k_class_s;  
            desc [dsc$b_dtype] = dsc$k_dtype_t;  
            desc [dsc$w_length] = (L);  
            desc [dsc$a_pointer] = (A);  
        END  
        %,  
  
    $str_desc_set (desci, L, A)     ! Copy new length and pointer fields (both static and dynamic)  
        =  
        BEGIN  
        BIND  
            desc = (desci) : BLOCK [, BYTE];  
            desc [dsc$w_length] = (L);  
            desc [dsc$a_pointer] = (A);  
        END
```

```
%.  
! And shorthand for just a descriptor declaration  
$desc_block  
= BLOCK [dsc$k_s_bln, BYTE]  
%.  
! Short form for byte vector reference  
$ref_bvector  
= REF $bvector  
%.  
! Short form for byte block reference  
$ref_bblock  
= REF $bblock  
%;  
STRUCTURE $bvector [I; N] =  
[N]  
($bvector+I)<0,8,0>;  
!  
+ SIGNAL_STOP a condition assuming no return. LIB$exch_signal_STOP is not  
supposed to return, but BLISS doesn't know this, so we block further  
flow here. This will generate better code for us.  
-_  
MACRO $exch_signal_stop []  
=  
BEGIN  
LINKAGE  
LNK = CALL : PRESERVE (0,1,2,3,4,5,6,7,8,9,10,11);  
EXTERNAL ROUTINE  
LIB$STOP : ADDRESSING_MODE (GENERAL) LNK NOVALUE;  
BUILTIN  
R0;  
LIB$STOP (%REMAINING);  
RETURN (.R0);  
END  
%;  
!  
+ SIGNAL a condition and return.  
-_  
MACRO $exch_signal_return (code)  
=
```

```
BEGIN  
LOCAL  
    temp;  
  
    temp = (code);           ! Need to avoid multiple calls, etc  
    SIGNAL (.temp            ! Perform the actual signal of the error  
            %IF %LENGTH GTR 1 %THEN ,%REMAINING %FI);  
    RETURN .temp  
  
END  
%;  
  
!+ SIGNAL a condition and continue.  
-_  
MACRO Sexch_signal (code)  
= SIGNAL ( (code)          ! Perform the actual signal of the error  
          %IF %LENGTH GTR 1 %THEN ,%REMAINING %FI)  
%;
```

+ Initialize a control block type and size fields. We do not depend on them being in the standard positions

- MACRO

 \$block_init (addr, prefix)

 = BEGIN
 BIND
 addr2 = (addr) : BLOCK [, BYTE];
 addr2 [%NAME (prefix,'\$w_size')] = %NAME ('exchblk\$w_',prefix);
 addr2 [%NAME (prefix,'\$b_type')] = %NAME ('exchblk\$b_',prefix);
 END
 %;

+ Check a control block type and size fields. Note that we depend on them being in the standard positions

- MACRO

 \$block_check (level, addr, prefix, error_code)

 = %IF switch_variant GEO (level)
 %THEN
 BEGIN
 EXTERNAL ROUTINE
 exch\$util_block_check : jsb_r0r1r2 NOVALUE;
 exch\$util_block_check ((addr), (error_code),
 (%NAME ('exchblk\$w_',prefix) ^ 16 OR %NAME ('exchblk\$k_',prefix)));
 END
 %FI
 %;

- MACRO

 \$block_check_if_nonzero (level, addr, prefix, error_code)

 = %IF switch_variant GEO (level)
 %THEN
 BEGIN
 BIND
 addr2 = (addr) : BLOCK [, BYTE];
 IF addr2 NEQ 0
 THEN \$block_check ((level), (addr), (prefix), (error_code));
 END
 %FI
 %;

+ Check for a logic error. If the expression is not true, then we have a problem.

- MACRO

 \$logic_check (level, condition, error_code)

 = ! See if a compile time check is possible

```
| XIF %CTCE ((condition))
| XTHEN
```

```
| The condition is a compile-time expression. There is one special case, when the
| condition is the string "(false)". This is used as an unconditional logic abort.
| If we have "(false)", then do a naked SIGNAL_STOP
```

```
| XIF %IDENTICAL (condition, (false))
| XTHEN
|     SIGNAL_STOP (exch$_badlogic, 1, (error_code))
```

```
| The condition is a normal test. If it is true, print a message that the condition
| was verified during compilation. If false, generate a serious error.
```

```
| XELSE
|     XIF (condition)
|     XTHEN
|         %PRINT ('assumption ',error_code,' verified during compilation')
|     XELSE
|         %ERROR ('assumption ',error_code,' is not true')
|     XFI
```

```
| The condition is not a compile-time constant. If the current variant calls for it,
| generate run-time code to test the assumption.
```

```
| XELSE
|     XIF switch_variant GEQ (level)
|     XTHEN
|         BEGIN
|             IF NOT (condition)
|             THEN
|                 SIGNAL_STOP (exch$_badlogic, 1, (error_code));
|             END
|         XFI
|     XFI
|     %;
```

+ Most messages are defined as warnings so that we can signal without changing flow of execution. Several macros are defined in EXCLIB to change the severity code. These are \$warning_stat, \$success_stat, \$error_stat, \$info_stat and \$severe_stat. A typical use would be:

```
status = lib$foo (bar);
IF NOT .status
THEN
  BEGIN
    $warning_stat (.status);      ! Convert unknown severity to warning
    SIGNAL (.status);
    RETURN .status;
  END;
```

These macros modify the status variable and return the value of the modified status.

For those situations where it is inappropriate to modify the code, forms of the macro are available which modify copies of the status code. These macro names have "_copy" appended to the modify form of the macro name. A couple of examples of their use are:

```

status = lib$foo (bar);
IF NOT .status
THEN
BEGIN
new_stat = $warning_status_copy (.status);
SIGNAL (.new_stat);
RETURN (.status);
END;
status = lib$foo (bar);
IF NOT .status
THEN
BEGIN
SIGNAL ($error_status_copy (exch$_badfoo), 0, .status);
RETURN (exch$_badfoo);
END;

```

Note that the "copy" forms have value arguments, the regular forms have address arguments.

Convert status codes to specific status values.

MACRO

```
$inhibit_msg_copy (status)
=
BEGIN
LOCAL
    status2 : BLOCK [4, BYTE];
status2 [0,0..32,0] = status;
status2 [sts$v_inhib_msg] = 1;
.status2
END
%.
```

```
Swarning_stat (status)
=
BEGIN
BIND
    status2 = status : BLOCK [4, BYTE];
    status2 [sts$v_severity] = sts$k_warning; ! Force status to warning
    .status2 ! Value of block is new code
END
%.
```

```
Swarning_stat_copy (status)
=
BEGIN
LOCAL
    status2 : BLOCK [4, BYTE];
    status2 [0,0,32,0] = status; ! Copy the whole code
    status2 [sts$v_severity] = sts$k_warning; ! Force status to warning
    .status2 ! Value of block is new code
END
%.
```

```
$success_stat (status)
=
BEGIN
BIND
    status2 = status : BLOCK [4, BYTE];
    status2 [sts$v_severity] = sts$k_success; ! Force status to success
    .status2 ! Value of block is new code
END
%.
```

```
$success_stat_copy (status)
=
BEGIN
LOCAL
    status2 : BLOCK [4, BYTE];
    status2 [0,0,32,0] = status; ! Copy the whole code
    status2 [sts$v_severity] = sts$k_success; ! Force status to success
    .status2 ! Value of block is new code
END
%.
```

```
Serror_stat (status)
=
BEGIN
BIND
    status2 = status : BLOCK [4, BYTE];
    status2 [sts$v_severity] = sts$k_error; ! Force status to error
    .status2 ! Value of block is new code
END
%.
```

```
Serror_stat_copy (status)
=
BEGIN
LOCAL
```

```
        status2 : BLOCK [4, BYTE];
status2 [0,0,32,0] = status;           ! Copy the whole code
status2 [sts$V_severity] = sts$K_error; ! Force status to error
.status2                                ! Value of block is new code
END
%.

$info_stat (status)
=
BEGIN
BIND
    status2 = status : BLOCK [4, BYTE];
status2 [sts$V_severity] = sts$K_info;   ! Force status to info
.status2                                ! Value of block is new code
END
%.

$info_stat_copy (status)
=
BEGIN
LOCAL
    status2 : BLOCK [4, BYTE];
status2 [0,0,32,0] = status;           ! Copy the whole code
status2 [sts$V_severity] = sts$K_info;   ! Force status to info
.status2                                ! Value of block is new code
END
%.

$severe_stat (status)
=
BEGIN
BIND
    status2 = status : BLOCK [4, BYTE];
status2 [sts$V_severity] = sts$K_severe; ! Force status to severe
.status2                                ! Value of block is new code
END
%.

$severe_stat_copy (status)
=
BEGIN
LOCAL
    status2 : BLOCK [4, BYTE];
status2 [0,0,32,0] = status;           ! Copy the whole code
status2 [sts$V_severity] = sts$K_severe; ! Force status to severe
.status2                                ! Value of block is new code
END
%.

! Special debug and trace macros
MACRO
$dbgtrc_prefix (string)          ! Declare a nested macro with the value of the string
=
MACRO $dbgtrc_prefix_string = string %QUOTE %
```

%.

```
Scheck_call (level, routine_addr)      ! Call the routine depending on variant level
= %IF switch_variant GEQ (level)
%THEN
  BEGIN
    EXTERNAL ROUTINE routine_addr : ADDRESSING_MODE (GENERAL);
    routine_addr (%REMAINING)
  END;
%FI
%;
```

```
; Message print routines
MACRO
$print_lit (string)
    = lib$put_output (%ASCII string)
    ;.

$trace_print_lit (string)
    = %IF switch_trace
      %THEN   lib$put_output (%ASCII %STRING ($dbgtrc_prefix_string, string))
      %FI ! switch_trace
    ;.

$debug_print_lit (string)
    = %IF switch_debug
      %THEN   lib$put_output (%ASCII %STRING ($dbgtrc_prefix_string, string))
      %FI ! switch_debug
    ;.

$print_desc (desc)
    = lib$put_output (desc)
    ;.

$trace_print_desc (desc)
    = %IF switch_trace
      %THEN
        BEGIN
          EXTERNAL ROUTINE exch$util_fao_buffer;
          lib$put_output (
            exch$util_fao_buffer (%ASCII %STRING ($dbgtrc_prefix_string, '!AS'), desc))
        END
      %FI ! switch_trace
    ;.

$debug_print_desc (desc)
    = %IF switch_debug
      %THEN
        BEGIN
          EXTERNAL ROUTINE exch$util_fao_buffer;
          lib$put_output (
            exch$util_fao_buffer (%ASCII %STRING ($dbgtrc_prefix_string, '!AS'), desc));
        END
      %FI ! switch_debug
    ;.

$print_fao (string)
    = BEGIN
```

```
EXTERNAL ROUTINE exch$util_fao_buffer;
lib$put_output (
    exch$util_fao_buffer (%ASCID string
        %IF %[LENGTH GTR 1 %THEN ,%REMAINING %FI))
END
%.

$trace_print_fao (string)
=%
%IF switch_trace
%THEN
    BEGIN
        EXTERNAL ROUTINE exch$util_fao_buffer;
        lib$put_output (
            exch$util_fao_buffer (%ASCID %STRING ($dbgtrc_prefix_string, string)
                %IF %[LENGTH GTR 1 %THEN ,%REMAINING %FI))
        END
    %FI ! switch_trace
%.

$debug_print_fao (string)
=%
%IF switch_debug
%THEN
    BEGIN
        EXTERNAL ROUTINE exch$util_fao_buffer;
        lib$put_output (
            exch$util_fao_buffer (%ASCID %STRING ($dbgtrc_prefix_string, string)
                %IF %[LENGTH GTR 1 %THEN ,%REMAINING %FI))
        END
    %FI ! switch_debug
%.
```

; Macros to manipulate queues

MACRO

; Initialize the header of a queue. This means make each of the 2 pointers in the header point to the header.

\$queue_initialize (q_header)

=

BEGIN

BIND

qh = (q_header) : VECTOR [2, LONG];

qh [0] = _qh_;

qh [1] = _qh_;

END

%.

; Insert an element at the head of a queue.

\$queue_insert_head (item, q_header)

=

BEGIN

BUILTIN

INSQUE;

BIND

qh = (q_header) : VECTOR [2, LONG];

INSQUE ((item), _qh_ [0])

END

%.

; Insert an element at the tail of a queue.

\$queue_insert_tail (item, q_header)

=

BEGIN

BUILTIN

INSQUE;

BIND

qh = (q_header) : VECTOR [2, LONG];

INSQUE ((item), _qh_ [1])

END

%.

+ Remove the indicated element from a queue. The first parameter is the address of the element. The second parameter is optional.

; If supplied, it is the address of a longword in which to store the element removed from the queue or 0 if no element was present in the queue. The value of the expression is TRUE if a element was removed from the

! queue and FALSE otherwise.

If the second parameter is not supplied, the value of the expression is the address of the element removed from the queue or 0 if no element was present in the queue.

```
-  
$queue_remove (q_element, element)  
= BEGIN  
  BIND _qhead_ = (q_element) : VECTOR [2, LONG];  
  BUILTIN REMQUE;  
  XIF (%NULL (element))  
  XTHEN LOCAL _T_ : REF VECTOR [2, LONG];  
  XELSE BIND _T_ = (element) : REF VECTOR [2, LONG];  
  XFI  
  IF (REMQUE (_qhead_, _T_))  
  THEN BEGIN  
    ! queue was empty  
    IF (%NULL (element))  
    THEN 0  
    ELSE (_T_ = 0; FALSE)  
  END  
  ELSE BEGIN  
    IF (%NULL (element))  
    THEN  
      _T_  
    ELSE true  
  END  
END  
;
```

* Remove an element from the head of a queue. The first parameter is the address of the queue header. The second parameter is optional.

If supplied, it is the address of a longword in which to store the element removed from the queue or 0 if no element was present in the queue. The value of the expression is TRUE if a element was removed from the queue and FALSE otherwise.

! If the second parameter is not supplied, the value of the expression is the address of the element removed from the queue or 0 if no element was present in the queue.

```
!-
$queue_remove_head (q_header, element)
=
BEGIN
BIND
  _qh_ = (q_header) : VECTOR [2, LONG];
%IF (%NULL (element))
%THEN
  $queue_remove (_qh_ [0])
%ELSE
  $queue_remove (_qh_ [0], element)
%FI
END
%.
```

+ Remove an element from the tail of a queue. The first parameter is the address of the queue header. The second parameter is optional.

If supplied, it is the address of a longword in which to store the element removed from the queue or 0 if no element was present in the queue. The value of the expression is TRUE if a element was removed from the queue and FALSE otherwise.

If the second parameter is not supplied, the value of the expression is the address of the element removed from the queue or 0 if no element was present in the queue.

```
!-
$queue_remove_tail (q_header, element)
=
BEGIN
BIND
  _qh_ = (q_header) : VECTOR [2, LONG];
%IF (%NULL (element))
%THEN
  $queue_remove (_qh_ [1])
%ELSE
  $queue_remove (_qh_ [1], element)
%FI
END
%.
```

! Test a queue for emptiness. TRUE if the queue is empty, FALSE if the queue is not empty

```
$queue_empty (q_header)
=
BEGIN
BIND
  _qh_ = (q_header) : VECTOR [2, LONG];
  _qh_ EQLA _qh_ [0]
```

EXCLIB.B32;1

16-SEP-1984 16:59:32.12 Page 17^{H 9}

END
X;

Literal definitions:

define literals for BLISS true and false values

LITERAL

```
true = 1
false = 0
;
```

Define values of some ASCII characters

LITERAL

NUL = 0,	null
LF = 10,	line feed
VT = 11,	vertical tab
FF = 12,	form feed
CR = 13,	carriage return
CTRLZ = 26,	control z
ESC = 27,	escape
DEL = 127,	rubout
;	

Define the Radix-50 equivalents for FILE.BAD

LITERAL

R50_EMPTY = %RAD50_11 'EMPTY ',	longword "EMPTY "
R50_FIL = %RAD50_11 'FIL',	word "FIL"
R50_FILE = %X '1F4026F4',	longword "FILE "
R50_BAD = %X '0CAC',	word "BAD"
R50_SYS = %X '7ABB',	word "SYS"
;	

Linkage definitions:

LINKAGE

jsb_r0r1 = JSB (REGISTER=0, REGISTER=1)	: NOPRESERVE(0,1) NOTUSED(2,3,4,5,6,7,8,9,10,11),
jsb_r0r1r2 = JSB (REGISTER=0, REGISTER=1, REGISTER=2)	: NOPRESERVE(0,1,2) NOTUSED(3,4,5,6,7,8,9,10,11),
jsb_r1 = JSB (REGISTER=1)	: NOPRESERVE(0,1) NOTUSED(2,3,4,5,6,7,8,9,10,11),
jsb_r1r2 = JSB (REGISTER=1, REGISTER=2)	: NOPRESERVE(0,1,2) NOTUSED(3,4,5,6,7,8,9,10,11),
jsb_r1r2r3 = JSB (REGISTER=1, REGISTER=2, REGISTER=3)	: NOPRESERVE(0,1,2,3) NOTUSED(4,5,6,7,8,9,10,11),
jsb_r2r3 = JSB (REGISTER=2, REGISTER=3)	: NOPRESERVE(0,1,2,3) NOTUSED(4,5,6,7,8,9,10,11),
jsb_r3r4 = JSB (REGISTER=3, REGISTER=4)	: NOPRESERVE(0,1,2,3) NOTUSED(4,5,6,7,8,9,10,11),
jsb_get = JSB (REGISTER=5, REGISTER=6, REGISTER=7)	: NOPRESERVE(0,1,2,3,4) NOTUSED(5,6,7,8,9,10,11),
jsb_put = JSB (REGISTER=9, REGISTER=10)	: NOPRESERVE(0,1,2,3,4,5,6,7,8,9,10) NOTUSED(11)

EXCLIB.B32;1

16-SEP-1984 16:59:32.12 Page 19

;

! Run-time library and other routines external to the facility

EXTERNAL ROUTINE

cli\$dc1_parse	: ADDRESSING_MODE (GENERAL),	! Command parsing routine
cli\$dispatch	: ADDRESSING_MODE (GENERAL),	! Action routine dispatch
cli\$get_value	: ADDRESSING_MODE (GENERAL),	! Entity value fetch
cli\$present	: ADDRESSING_MODE (GENERAL),	! Entity presence boolean
lib\$find_file	: ADDRESSING_MODE (GENERAL),	! Wildcard files-11 processing
lib\$free_vm	: ADDRESSING_MODE (GENERAL),	! Releases memory
lib\$get_input	: ADDRESSING_MODE (GENERAL),	! Get a line from SYSSINPUT
lib\$get_vm	: ADDRESSING_MODE (GENERAL),	! Gets memory
lib\$put_output	: ADDRESSING_MODE (GENERAL),	! Display a line on SYSSOUTPUT
ots\$cvt_t_l	: ADDRESSING_MODE (GENERAL),	! ASCII decimal to longword
ots\$cvt_to_l	: ADDRESSING_MODE (GENERAL),	! ASCII octal to longword
ots\$cvt_tz_l	: ADDRESSING_MODE (GENERAL),	! ASCII hexadecimal to longword
str\$copy_dx	: ADDRESSING_MODE (GENERAL),	! Copy string of any class
str\$freeT_dx	: ADDRESSING_MODE (GENERAL)	! Release dynamic string

:

! Define the lengths of control blocks here - Many of these need to be adjusted by system block sizes, so it can't be completely done in the SDL definition.

LITERAL

! An SEXCG is the global environment for the facility, the SDL block plus two RMS work areas

exchblk\$\$_excg = excg\$k_length + 2*(fab\$k_bln + rab\$k_bln + nam\$k_bln + (2*nam\$e_maxrss)),

! An SRMSB describes an RMS file, the SDL block plus one RMS work area

exchblk\$\$_rmsb = rmsb\$k_length + fab\$k_bln + rab\$k_bln + nam\$k_bln + (2*nam\$e_maxrss),

! A SVOLB contains the structures for a volume, the SDL block plus one RMS work area

exchblk\$\$_volb = volb\$k_length + fab\$k_bln + rab\$k_bln + nam\$k_bln + (2*nam\$e_maxrss),

! The following don't need adjusting, but we want to keep all the EXCHBLK\$\$_ definitions in one place

exchblk\$\$_copy = copy\$k_length,	! Size of the work area for the COPY command
exchblk\$\$_dire = dire\$k_length,	! Size of the work area for the DIRECTORY command
exchblk\$\$_dos11 = dos11\$k_length,	! Size of the DOS-11 specific extension to the volb
exchblk\$\$_dos11ctx = dos11ctx\$k_length,	! Size of the DOS-11 file context block
exchblk\$\$_filb = filb\$k_length,	! A \$FILB is a structure which describes an open file
exchblk\$\$_init = init\$k_length,	! Size of the work area for the INIT command
exchblk\$\$_namb = namb\$k_length,	! A \$NAMB is a structure which stores a fully parsed file name
exchblk\$\$_moun = moun\$k_length,	! Size of the work area for the MOUNT command
exchblk\$\$_rt11 = rt11\$k_length,	! Size of the RT-11 specific extension to the volb
exchblk\$\$_rt11ctx = rt11ctx\$k_length,	! Size of the RT-11 file context block
exchblk\$\$_rtnam = rtnam\$k_length	! Size of the work area for the DIRECTORY command

:

; Message codes defined in SRCS:EXCMSG.MSG

EXTERNAL LITERAL

exch\$_accessfail,	failed to access volume (\$GETDVI service failure)
exch\$_badfilename,	File name not valid for given volume
exch\$_badlogic,	Adds error number to shared message
exch\$_badpad,	Improper /RECORD FORMAT=PAD option
exch\$_binchksum,	Bad formatted binary record
exch\$_binrecfmt,	Bad formatted binary record
exch\$_blockcheck,	Block check failed
exch\$_blockcheck0,	Block check failed because block address is 0
exch\$_canceled,	Command canceled
exch\$_closeerr,	Error closing file
exch\$_closeforeign,	Error closing foreign device
exch\$_copied,	Log message for copy command
exch\$_copyboot,	Log message for copy /boot command
exch\$_copnewname,	File copied with new name
exch\$_createvirt,	Error creating virtual volume
exch\$_deleted,	Deleted copy of a file
exch\$_deleteprev,	Deleted previous copy of a file
exch\$_devonly,	Device spec only, other parts of file name ignored
exch\$_devnotsuit,	Device is not suitable for EXCHANGE
exch\$_dire_error,	Error writing directory
exch\$_dismounted,	Device has been dismounted
exch\$_dos11_badlabel,	Invalid label found on dos11 tape
exch\$_dos11_blocksize,	Invalid block (>512 bytes) found on dos11 tape
exch\$_dos11_ioerror,	Error during I/O on dos11 tape
exch\$_dos11_position,	Rewinding tape to find correct position
exch\$_filenotfound,	Unable to locate file
exch\$_fill1_norec,	No /RECORD for files-11
exch\$_ignore_dire,	Ignoring directory specification
exch\$_ignore_vers,	Ignoring file version number
exch\$_illmtcopy,	Illegal magtape copy, input and output on same device
exch\$_initialized,	Device has been initialized
exch\$_invrecfmt,	Record format not valid for volume type
exch\$_invvolfmt,	Volume format not valid for operation
exch\$_many_to_one,	Multiple input files were given but only one output file
exch\$_mounted,	Volume mounted (success)
exch\$_mounterror,	Error performing VMS \$mount service
exch\$_mountvir,	Virtual volume mounted (success)
exch\$_noalloc,	/ALLOCATE ignored on tape output
exch\$_nocarriage,	/CARRIAGE ignored on output
exch\$_nocopbad,	Couldn't create, .BAD file with wildcarded names
exch\$_nocopbaddel,	Couldn't create, have to delete .BAD file
exch\$_nocopdup,	Couldn't create, already created same name
exch\$_nocoplock,	Couldn't create, volume is writelocked
exch\$_nocopnodef,	Couldn't create, file of same name and /NODELETE given
exch\$_nocopprot,	Couldn't create, file of same name protected against modification
exch\$_nocopsamdev,	Illegal copy to same device
exch\$_nocopsysdel,	Illegal copy of .SYS when existing .SYS present
exch\$_nocopyboot,	Unable to copy boot info
exch\$_nodellock,	File not deleted, volume locked
exch\$_nodevice,	Device spec missing
exch\$_noremove,	Device spec cannot have node field
exch\$_norendev,	Illegal rename to different device
exch\$_norenexists,	Not renamed, already exists

M 9
! exch\$_norenlock, ! Files not renamed, volume locked
exch\$_nosysact, No action on .SYS files
exch\$_notcopied, File not copied
exch\$_notcop_retry, File not copied, will retry
exch\$_notdeleted, ! File not deleted
!\\ exch\$_notimplement, ! Feature not yet implemented
exch\$_notmounted, Device is not mounted on EXCHANGE
exch\$_notsamedev, Input and output not same device (copy /boot)
exch\$_notvallen, /REC=LEN requires FIXED
exch\$_novolumes, No volumes are mounted
exch\$_openforeign, Open failed on a foreign volume
exch\$_openvirtual, Open failed on a virtual volume
exch\$_opnotperdos, Operation not permitted on DOS-11 volume
exch\$_opnotperf11, Operation not permitted on Files-11 volume
!\\ exch\$_opnotperrt11, Operation not permitted on RT-11 volume (not yet needed)
!\\ exch\$_opnotperrtmt, Operation not permitted on RT-11 magtape volume
exch\$_parseerr, Bad file parameter syntax
exch\$_partcopied, File partially copied
exch\$_readcheck, Error detected during read check
exch\$_readcheckrec, Error detected during read check was recovered
exch\$_readerrrec, Error detected during read was recovered
exch\$_recover, Directory recovery message
exch\$_rectoobig, Bad formatted binary record
exch\$_renamed, File renamed log message
exch\$_rt11_baddirct, RT-11 directory error
exch\$_rt11_badfile, Bad block file created
exch\$_rt11_bigbadfile, Bad block file contains some good blocks
'exch\$_rt11_dirsize, Device size disagrees with directory size
exch\$_rt11_errlock, RT-11 directory error
exch\$_rt11_extra, Too many extra words
exch\$_rt11_noend, RT-11 directory error
exch\$_rt11_overflow, RT-11 directory error
exch\$_rt11_stblock, RT-11 directory error
exch\$_rt11_toomanyblk, RT-11 directory error
exch\$_rt11_toomanyseg, RT-11 directory error
exch\$_rt11_unkent, RT-11 directory error
exch\$_rtouteof, End of file on output file
exch\$_rtprotect, File protected against modification
exch\$_stmrecfmt, Bad stream record format
exch\$_stnotavail, Start block not available
exch\$_strtnomulti, Can't say /START with multiple input files
exch\$_toomanycol, Too many columns requested
exch\$_trace, Header for a status trace
exch\$_typed, Log message for type command
exch\$_virtnochage, Cannot change size of virtual devices
exch\$_vmsmount, Volume has been mounted on VMS
exch\$_volmount, Volume is already mounted
exch\$_volume_full, Output volume is full
exch\$_waiterr, Error waiting for RMS operation
exch\$_writecache, Writing modified directory segments
exch\$_writecheck, Error detected during write check
exch\$_writecheckrec, Error detected during write check was recovered
exch\$_writeerrrec, Error detected during write was recovered
exch\$_writelock, ! Volume is write-locked
:

```
! Shared message definitions
```

```
$shr msgdef
  texch, 248, local,
  (badlogic, warning),
  (badvalue, warning),
  (closeout, warning),
  (confqual, warning),
  (insvirmem, warning),
  (openin, warning),
  (openout, warning),
  (readerr, warning),
  (writeerr, warning)
);
```

```
$shr msgdef
  {msg, 3, local,
   (syntax, severe)
};
```

```
! Other symbols which need explicit declarations
```

```
EXTERNAL LITERAL
```

```
cli$_comma,           ! Parameter ended with a comma
cli$_concat,          ! Parameter ended with a plus sign
cli$_locneg,          ! An explicit /NOqual for local qual
cli$_locpres,         ! An explicit /qual for local qual
cli$_nocomd,          ! CLI saw a blank line and burped
cli$_negated,         ! An explicit /NOqual was given
cli$_present,         ! An explicit /qual was given
cli$_facility;        ! CLI facility code
```

```
! Storage external to all modules
```

```
EXTERNAL
```

```
exch$cld_table : ADDRESSING_MODE (LONG_RELATIVE)           ! Command table for CLISDCL_PARSE
;
```

```
! External data - defined in EXCH$MAIN module
```

```
EXTERNAL
```

```
exch$qq_dyn_str template : $desc_block ADDRESSING_MODE (LONG_RELATIVE),
exch$sa_gbl : REF BLOCK [,BYTE] ADDRESSING_MODE (LONG_RELATIVE)      ! An initialized, null dynamic string descriptor
;                                         ! The pointer to the known world
```

```
! END
! ELUDOM
```

```
! End of module EXCLIB
```

0159 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

USSLINK
COM

EXCDEFS
5DL

MATLUAF
COM

USSTSLNK
COM

EXCHNG

XATEST
COM

EXCHANGE
MAP

LABIO
OPT

MSCPMOUNT LABIOCIN
GOM OPT

EXCREQ
R32